On-line Monitoring Framework

Double-blind
(double-blind)

Abstract

I/O monitoring is important to find performance inefficiencies in applications. Most of the monitoring tools are based on the following methods. Injection of instrumentation code into applications is an intrusive method which requires re-compilation of the application. Collecting I/O data from proc-files provide no information about file access. Dynamic linking of instrumentation libraries with `LD_PRELOAD` is typically used for creation of trace files for the post-mortem analysis.

In our work, we construct an HPC on-line monitoring framework on top of open source software: SIOX, FUSE, Elasticsearch and Grafana. This framework collects I/O statistics from applications and mount points. The latter can be used for a non-intrusive monitoring of virtual memory allocated with mmap().

Instead of gathering I/O statistics from global system variables, like many other monitoring tools do, in our approach statistics come directly from I/O interfaces POSIX, MPI, HDF5 and NetCDF. For interception of I/O calls we exploit `LD_PRELOAD` feature.

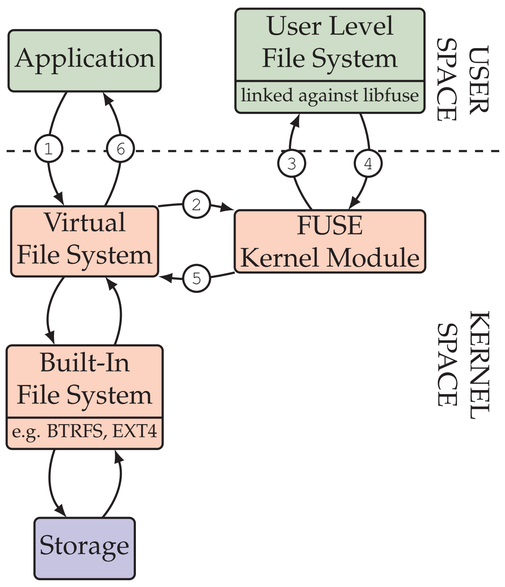
SIOX

SIOX is modular performance analysis framework published under LGPL. For I/O interception it exploits `LD_PRELOAD`.

- Support of several I/O interfaces
  - MPI, POSIX, HDF5 and NetCDF4
- On-line analysis
  - I/O analysis during program execution
- Off-line analysis
  - I/O analysis after program termination

FUSE/IOFS

FUSE (File system in user space) allows us to create files systems that can be run in non-privileged mode.



IOFS is an auxiliary tool that implements the FUSE interface. The purpose is to mount a directory of built-in file system to user defined mount point. Since IOFS has no cache, it redirects all I/O accessed from mount point to the directory.

Elasticsearch

Elasticsearch is a distributed, scalable, real-time search and analytics engine, published under the Apache 2 license.

- Indexing of all field allow very fast lookups, and makes it real-time capable

Grafana

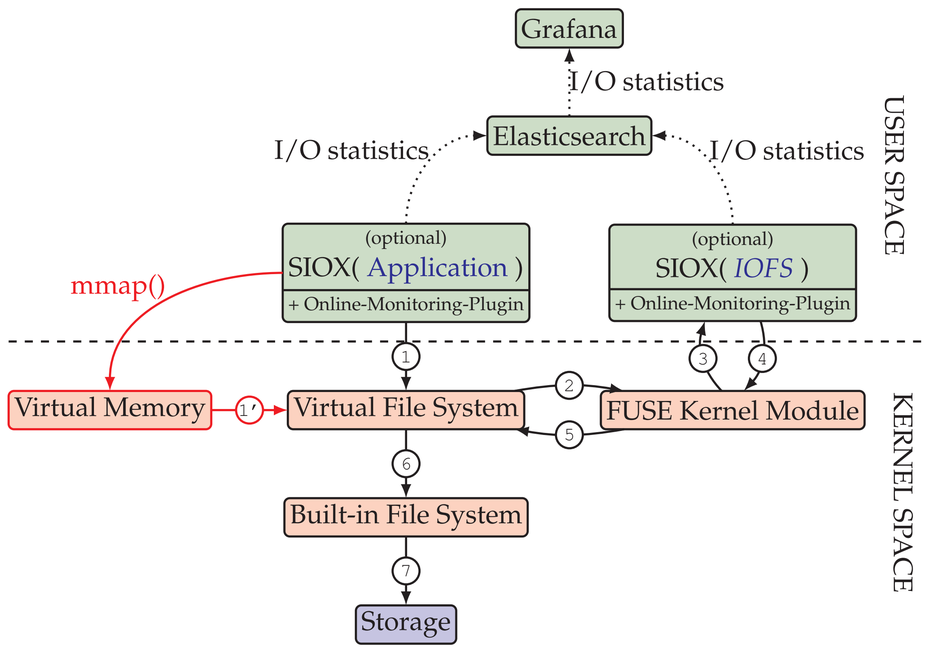
Grafana is a feature-rich, interactive visualization and dashboard software. For visualization, it provides different widgets, e.g., time series, tables, text fields for single metrics.

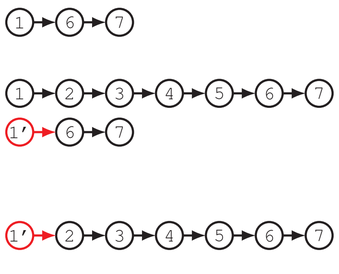
- Rich graphing: Interactive, editable graphs; Multiple Y-axes, Logarithmic scales and options
- Mixed Styling: Mix lines, points and bars; Mix stacked with isolated series
- Template Variables; Variables are automatically filled with values from DB
- Repeating Panels: Automatically repeat rows or panels for each selected variable value
- Annotations: Show events from data sources in the graphs

Framework Architecture

The on-line monitoring plug-in captures data from multiple sources and sends it in aggregated form in adjustable time intervals to Elasticsearch. Statistics can be visualized by Grafana.

- **SIOX(Application)** intercepts I/O operation calls from application
- **SIOX(IOFS)** intercepts I/O operation calls from mount point. The name of application and process number are lost.





Direct file I/O path. Typically, supported by most instrumentation tools.

Redirected file I/O path over FUSE.

Direct mmap I/O path. Problematic path, where instrumentation of I/O operations as a non-privileged user is not possible, because virtual memory lies in kernel space.

Redirected mmap I/O path. In our novel approach the redirection of I/O from kernel to user space allows SIOX to intercept the I/O calls within elevated privileges.

I/O Statistics

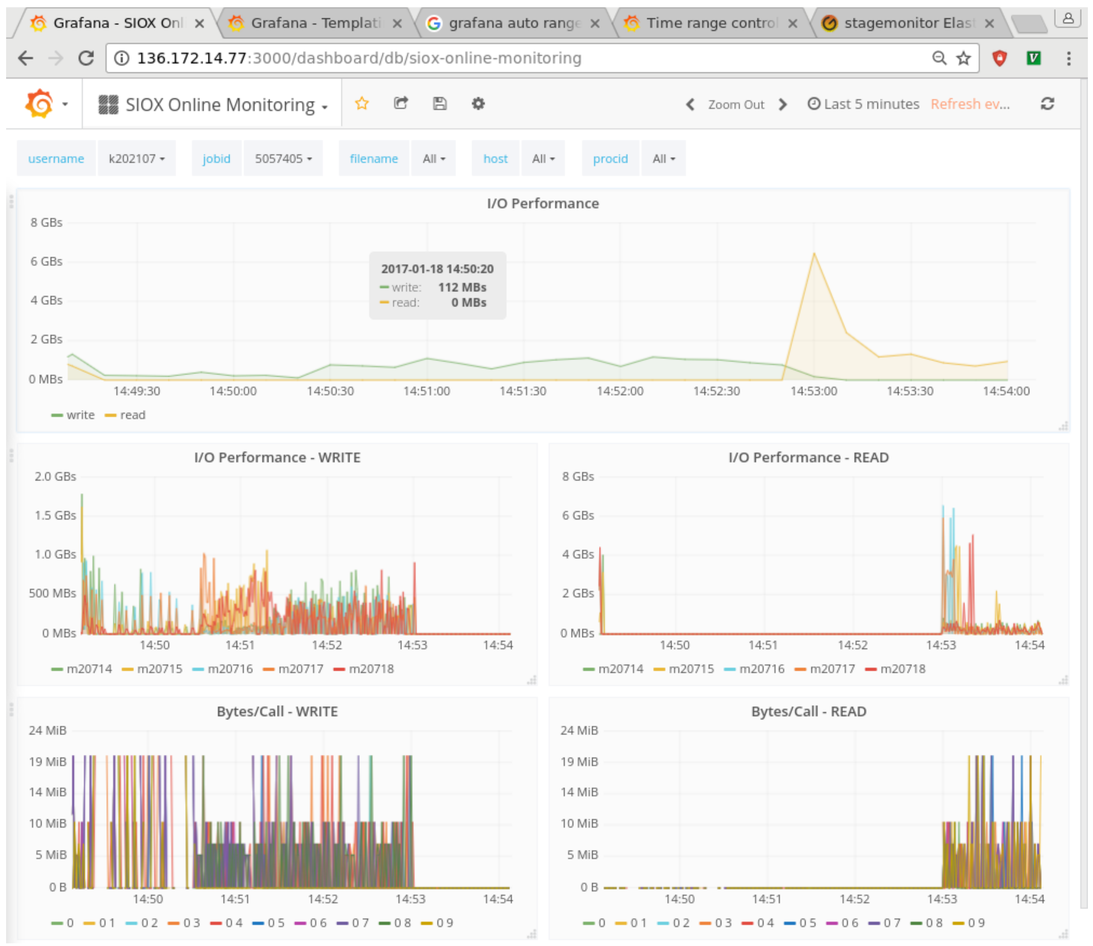
The table shows the current set of captured I/O statistics, incl. type and source. Typification is based on usage in Grafana.

- Metrics are aggregated over a time interval to reduce the amount of data.
- I/O operations are counted during the time interval.
- The value of `procid` is lost when monitoring a mount point.

Name	Type	Value
`write_duration`	metric (basic)	time spent for writing
`write_bytes`	metric (basic)	bytes written
`write_calls`	metric (basic)	number of I/O operations
`write_bytes_per_call`	metric (derived)	`write_bytes, write_calls`
`read_duration`	metric (basic)	time spent for reading
`read_bytes`	metric (basic)	bytes read
`read_calls`	metric (basic)	number of I/O operations
`read_bytes_per_call`	metric (derived)	`read_bytes, read_calls`
`filename`	tag	I/O operations
`access`	tag	I/O operations
`username`	tag	`SLURM_USER`
`hostname`	tag	`HOSTNAME`
`procid`	tag	`SLURM_PROCID`
`jobid`	tag	`SLURM_JOBID`
`layer`	tag	user defined
`timestamp`	date	system clock

Grafana Web-Interface

The web-interface shows I/O statistics about the selected job in near real-time. The interactive control allows a fast and convenient filtering of data. The default dashboards contain common graphs. Special dashboards can be easily created.



Usage of I/O statistics: on **y-axis:** we use metrics, on **x-axis:** we use time stamp and for **filtering:** we use tags.

Scalability

Metrics were generated by 200 processes (10 nodes x 20 processes/node). They were sent to Elasticsearch in bundles (100 metrics/bundle).

- Metrics generators were run on:
  - Intel Xeon E5-2695 v4 (Broadwell)
  - 2x 18-core @ 2.1GHz
  - FDR Infiniband
- Elasticsearch was deployed on an ordinary office computer:
  - Intel i7-6700 CPU (Skylake)
  - 4 cores @ 3.40GHz
  - Ethernet 1GB/s

Elasticsearch could receive about **750,000** metrics per second.

Overhead

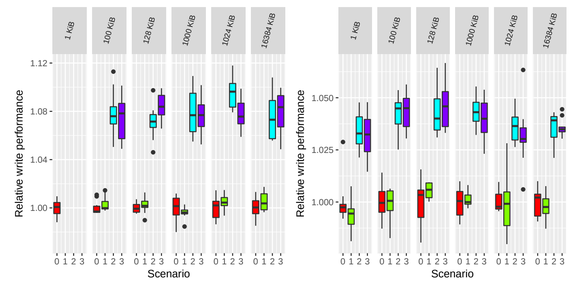
Experiments with different scenarios (0-3) are conducted with IOR and IOZone.

- **IOR** was used to perform file I/O.
- **IOZone** was used to perform mmap I/O.

For the measurement of overhead, we run a series of experiments on system equipped with:

- Intel Core i5-660, 4M Cache, 3.33 GHz
- 12 GB DDR3 RAM
- 2 TB HDD (Test disk)
- 500 GB HDD (OS disk)
- 1 GB/s network

For each blocksize the overhead is measured 10 times. All experiments are run on 1 node and 1 process per node. The size of the test file is 4 GiB. Scenario (0) is run without monitoring to get reference value. In other scenarios we monitor (1) application, (2) mount point, (3) and both.

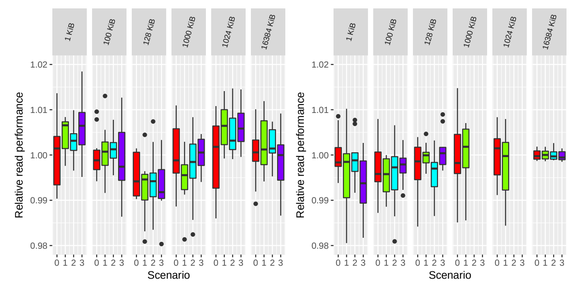


(a) IOR **(b)** IOZone

Fig. 1: Write overhead.

Outliers for 1 KiB

Scenario	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
1	1.125	1.133	1.139	1.137	1.142	1.147
2	3.506	3.537	3.580	3.590	3.652	3.662
3	4.738	4.888	5.120	5.078	5.257	5.384



(a) IOR **(b)** IOzone

Fig. 2: Read overhead.

Outliers for 1000 KiB

Scenario	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
2	1.224	1.241	1.261	1.257	1.271	1.288
3	1.250	1.257	1.260	1.265	1.267	1.300

Outliers for 1024 KiB

Scenario	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
2	1.260	1.282	1.288	1.293	1.308	1.342
3	1.270	1.275	1.288	1.287	1.298	1.304

Summary

- Built completely on open source software
- All components are highly scalable
- Operates fully in user space
- Provides support for
  - file I/O and mmap I/O
- Non-intrusive monitoring
  - No changes in application are required
- Intuitive web interface
  - Elaborated filtering functionality
  - Interactive and convenient control

Future Work

- In the next step we plan to deploy our framework in real HPC environment and run some experiments on real applications.
- Furthermore, we will be looking for ways to improve the user experience.